Computational Statistics: Lab 4

Group 03
Mohsen Pirmoradiyan, Ahmed Alhasan, Yash Pawar

22 Feb 2020

Question 1: Computations with Metropolis-Hastings

Consider the following probability density function:

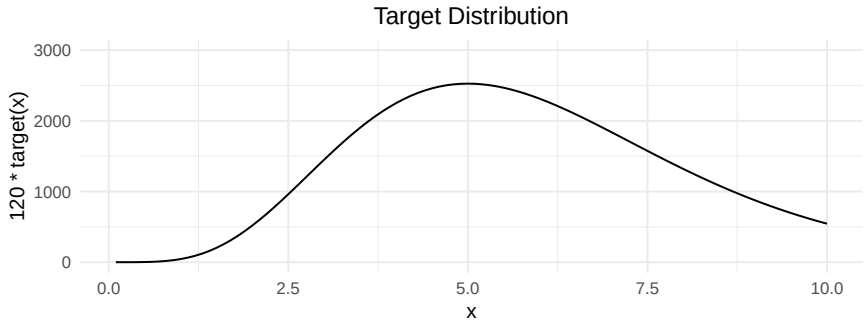
$$f(x) \propto x^5 e^{-x}, \quad x > 0$$

You can see that the distribution is known up to some constant of proportionality. If you are interested (NOT part of the Lab) this constant can be found by applying integration by parts multiple times and equals 120.

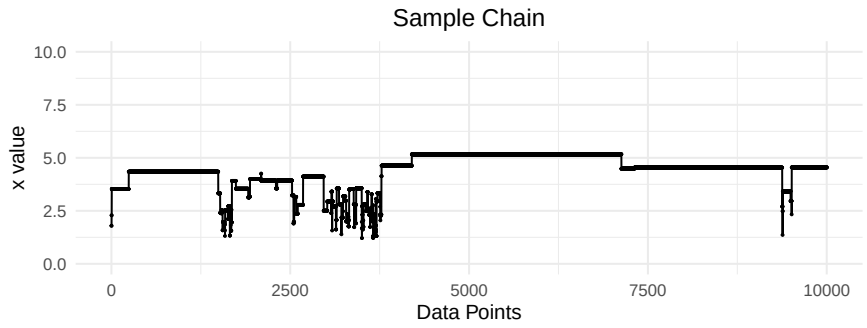
1. Use Metropolis-Hastings algorithm to generate samples from this distribution by using proposal distribution as log-normal $LN(X_t, 1)$, take some starting point. Plot the chain you obtained as a time series plot. What can you guess about the convergence of the chain? If there is a burn-in period, what can be the size of this period?

```
RNGversion(min(as.character(getRversion()), "3.6.2"))
library(ggplot2)
target <- function(x){
  x^5 * exp(-x)
}

proposed <- function(x, mean){
  dlnorm(x, meanlog = mean, sdlog = 1)
}

MH <- function(x0, n, prop){
  x    <- rep(0,n)
  x[1] <- x0
  for(i in 1:n){
    y     <- rlnorm(1,x[i],1)
    u     <- runif(1)
    alpha <- min(1, (target(y)/target(x[i])) * (prop(x[i], y)/prop(y, x[i])))
    ifelse(u < alpha, x[i+1] <- y, x[i+1] <- x[i])
  }
  return(x)
}
set.seed(12345, kind="Mersenne-Twister", normal.kind="Inversion")
res <- MH(x0 = rlnorm(1), n = 10000, prop = proposed)
```

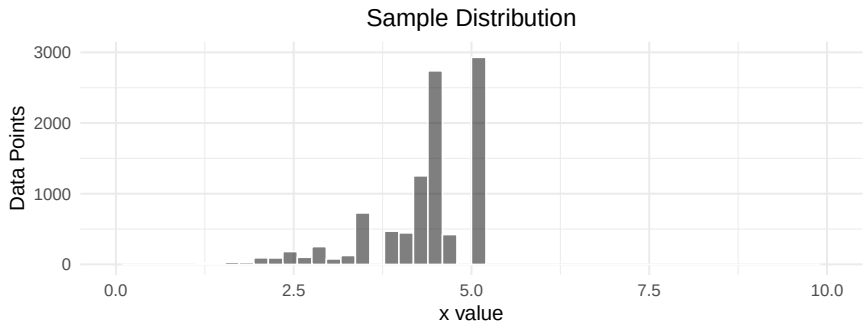
```
x <- seq(from = 0.1, to = 10, by = 0.1)
ggplot(as.data.frame(x), aes(x = x))+
  geom_line(aes(y = 120 * target(x)))+
  ylim(c(0,3000))+
  ggtitle("Target Distribution")+
  theme_minimal()+
  theme(plot.title = element_text(hjust=0.5))
```




```
myplot <- function(data){
  g1 <- ggplot(as.data.frame(data), aes(x = 1:length(data), y = as.numeric(data)))+
    geom_point(size = 0.3)+
    geom_line()+
    ylab("x value")+
    xlab("Data Points")+
    ylim(c(0,10))+
    ggtitle("Sample Chain")+
    theme_minimal()+
    theme(plot.title = element_text(hjust=0.5))

  g2 <- ggplot(as.data.frame(data), aes(x = as.numeric(data)))+
          geom_histogram(bins = 50,
                         alpha = 0.5,
                         color = "white",
                         fill = "black",
                         size = 0.4)+
          xlim(c(0,10))+
          ylim(c(0,3000))+
          ylab("Data Points")+
          xlab("x value")+
          ggtitle("Sample Distribution")+
          theme_minimal()+
          theme(plot.title = element_text(hjust=0.5))

  list(g1, g2)
}
myplot(res)[[1]]
```



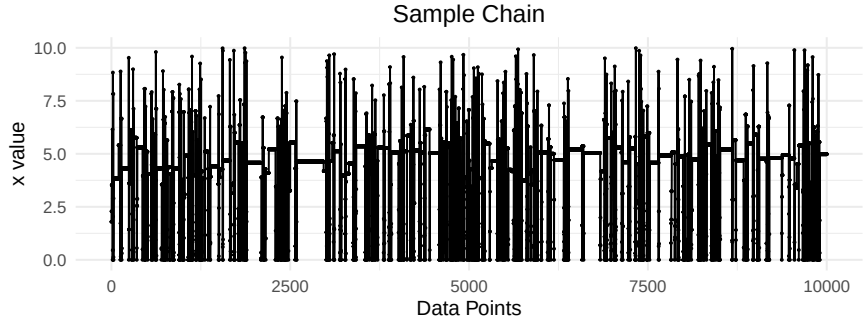

```
myplot(res)[[2]]
```



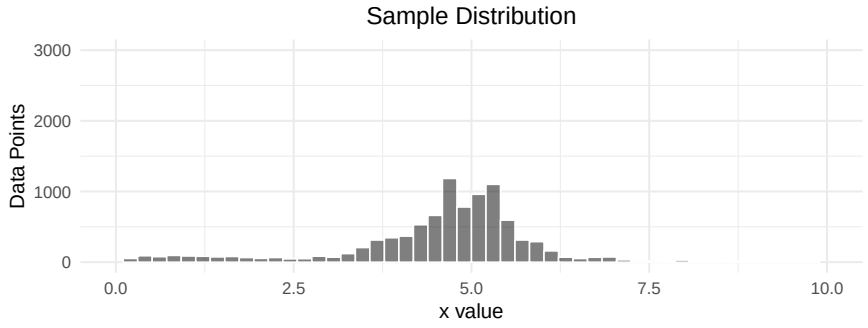
- The burn in period depends on the number of iterations/sample points (how much compuation we can afford) and on the starting point which will dictate how many sample points are in a low or high probablity region of the target distribution
- In this case since the the starting point is around x = 2 which is at the edge of the high probability region a burn in period of 2-3% is sufficient

2. Perform Step 1 by using the chi-square distribution $\chi^2([X_t + 1])$ as a proposal distribution, where [x] is the floor function, meaning the integer part of x for positive x, i.e. $[2.95] = 2$

```
proposed_2 <- function(x, df){
  dchisq(x, floor(df+1))
}

set.seed(12345, kind="Mersenne-Twister", normal.kind="Inversion")
res_2 <- MH(x = rlnorm(1), n = 10000, prop = proposed_2)
myplot(res_2)[[1]]
```



```
myplot(res_2)[[2]]
```



3. Compare the results of Steps 1 and 2 and make conclusions.

- The Chi-squared is much better proposed distribution than log-normal because log-normal has it's high probability region near zero while the equilibirium distribution has it's high probability region near 5 that cause a lot of the sampling iterations to fail the condition where we see the chain plot stuck at 5 or close to it for long time because once it gets there it is difficult for the algorithm to find better sample points

$$A = min(1, \frac{f(x.proposed)q(x.current|x.proposed)}{f(x.current)q(x.proposed|x.current)})$$

4. Generate 10 MCMC sequences using the generator from Step 2 and starting points 1,2,. . . , or 10. Use the Gelman-Rubin method to analyze convergence of these sequences.

```
set.seed(12345, kind="Mersenne-Twister", normal.kind="Inversion")
res_3 <- sapply(1:10, function(x0) MH(x = x0, n = 10000, prop = proposed_2))
library("coda")
mc_list <- mcmc.list()
for(i in 1:10){
  mc_list[[i]] <- as.mcmc(res_3[[i]])
}
gelman.diag(mc_list)
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

The Gelman_Rubin Covergence diagnostic evaluates the convergence by analyzing the difference between multiple Morkov chains.

Suppose we have estimated k sequences(k chains with different start points) of length n.

In this method the within-chain variance for each sequence is estimated, denoted by S_i where i denotes the i^{th} chain, from which the average within variance(W) is calculated: $W = \frac{1}{k}\sum_{i=1}^{k} {s_i}^2$.

The between-chains variances are estimated as well, denoted by B, which simply is a measure of differences between the different sequences(chains) having different start point.

Then, overall variance is estimated:

$$V = \frac{n-1}{n}W + \frac{1}{n}B$$

Finally the Gelman-Rubin factor is calculated as:

$$\sqrt{R} = \sqrt{\frac{V}{W}}$$

The values much larger than 1 indicate lack of convergence.

For computing this factor we used the function *gelman.diag*() from the package *coda*.

5. Estimate

$$\int_0^\infty xf(x)dx$$

5.1) Using samples from step 1

Given f(x) as the density function, this integration estimates the expected value of $f(x)$ which is the mean of sample over the generated series which is:

$$\frac{1}{n}\sum_{i=1}^{n} x.lnorm_i$$

```
mean(res)
```

```
## [1] 4.379969
```

5.2) Using samples from step 2

Given f(x) as the density function, this integration estimates the expected value of $f(x)$ which is the mean of sample over the generated series which is:

$$\frac{1}{n}\sum_{i=1}^{n} x.chi_i$$

```
mean(res_2)
```

```
## [1] 4.369632
```

6. The distribution generated is in fact a gamma distribution. Look in the literature and define the actual value of the integral. Compare it with the one you obtained.

The Gamma distribution is as follows:

$$Gamma(\alpha, \beta) = f(x|\alpha, \beta) = [\frac{1}{\Gamma(\alpha)\beta^\alpha}]x^{\alpha-1}exp(-x/\beta)$$

where α and β are called "shape factor" and "rate factor" respectively. The expected value of a Gamma distribution is the multiplication of these two factors: $\alpha\beta$.

In our case $\alpha = 6$ and $\beta = 1$ which yields the expected value of 6 for $f(x)$. Our estimation from step 2 is very close to this value which indicates that the chi-square distribution was a good proposal function for sampling from this distribution.

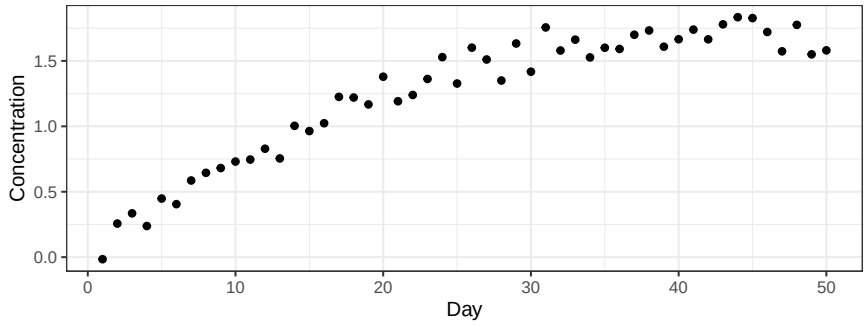
Question 2: Gibbs sampling

A concentration of a certain chemical was measured in a water sample, and the result was stored in the data chemical.RData having the following variables: - X: day of the measurement - Y: measured concentration of the chemical.

The instrument used to measure the concentration had certain accuracy; this is why the measurements can be treated as noisy. Your purpose is to restore the expected concentration values.

1. Import the data to R and plot the dependence of Y on X. What kind of model is reasonable to use here?

```
chemic = load("C:/Users/WizzCon/Desktop/Machine Learning/1. Workshop/6. Computational Statistics/1. Labs
chemic = data.frame(Day = X, Concentration = Y)

ggplot(chemic, aes(Day, Concentration))+geom_point()+theme_bw()
```



As the figure suggests a 2-degree polynomial or a quadratic function seems to well fit data.

2. A researcher has decided to use the following (random{walk) Bayesian model (n=number of observations, $\vec{\mu} = (\mu_1, ..., \mu_n)$ are unknown parameters):

$$Y_i \sim \mathcal{N}(\mu_i,\ variance = 0.2), \quad i = 1, ..., n$$

where the prior is

$$p(\mu_1) = 1$$

$$p(\mu_{i+1} \mid \mu_i) = \mathcal{N}(\mu_i, 0.2), \quad i = 1, ..., n1$$

Present the formulae showing the likelihood $p(\vec{Y} \mid \vec{\mu})$ and the prior $p(\vec{\mu})$. **Hint**: a chain rule can be used here $p(\vec{\mu}) = p(\mu_1)p(\mu_2 \mid \mu_1)p(\mu_3 \mid \mu_2)...p(\mu_n \mid \mu_{n1})$

Implementing the random-walk Bayesian model, present the formulae showing the likelihood $p(\vec{Y}|\vec{\mu})$ and the prior $p(\vec{\mu})$.

$$n = number\ of\ observations$$

$$\vec{\mu} = (\mu_1,\ \mu_2, ...,\ \mu_n) :\ unkonown\ parametesr$$

$$Y_i = N(\mu_i,\ variance = 0.2), \quad i = 1, ..., n$$

Where the prior is:

$$p(\mu_1) = 1$$
$$p(\mu_{i+1}|\mu_i) = N(\mu_i,\ 0.2), \quad i = 1, ..., n-1$$

The likelihood for $p(\vec{Y}|\vec{\mu})$ may be derived as:

$$L = \prod_{i=1}^{n} p(Y_i|\mu_i) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(Y_i - \mu_i)^2}{2\sigma^2})$$

$$= (\frac{1}{\sigma\sqrt{2\pi}})^n \exp(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (Y_i - \mu_i)^2), \qquad (1)$$

The prior function $p(\vec{\mu})$ may be derived as:

$$p(\vec{\mu}) = p(\mu_1)p(\mu_2|\mu_1)p(\mu_3|\mu_2)...p(\mu_n|\mu_{n-1})$$

Given $p(\mu_1) = 1$, we can conclude that:

$$p(\vec{\mu}) = \prod_{i=1}^{n-1} p(\mu_{i+1}|\mu_i)$$

We know that $p(\mu_{i+1}|\mu_i) = N(\mu_i,\ 0.2), \quad i = 1, ..., n-1$, so:

$$p(\vec{\mu}) = \prod_{i=1}^{n-1} p(\mu_{i+1}|\mu_i) = \prod_{i=1}^{n-1} \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(\mu_{i+1} - \mu_i)^2}{2\sigma^2})$$

$$= (\frac{1}{\sigma\sqrt{2\pi}})^{n-1} \exp(-\frac{1}{2\sigma^2} \sum_{i=1}^{n-1} (\mu_{i+1} - \mu_i)^2), \qquad (2)$$

(1) is the likelihood function for $p(\vec{Y}|\vec{\mu})$, and (2) is the prior function.

3. Use Bayes' Theorem to get the posterior up to a constant proportionality, and then find out the distributions of $(\mu_i \mid \vec{\mu}_{-i}, \vec{Y})$, where $\vec{\mu}_{-i}$ i is a vector containing all μ values except of μ_i

- Hint A:

Consider for separate formulae for $(\mu_1 \mid \vec{\mu}_{-1}, \vec{Y}), (\mu_n \mid \vec{\mu}_{-n}, \vec{Y})$ and then a formula for all remaining $(\mu_i \mid \vec{\mu}_{-i}, \vec{Y})$).

- Hint B:

$$\exp\left(-\frac{1}{d}\left((x-a)^2 + (x-b)^2\right)\right) \propto \exp\left(-\frac{(x-(a+b)/2)^2}{d/2}\right)$$

- Hint C:

$$\exp\left(-\frac{1}{d}\left((x-a)^2+(x-b)^2+(x-c)^2\right)\right) \propto \exp\left(-\frac{(x-(a+b+c)/3)^2}{d/3}\right)$$

The Baye's theorem:

$$p(\theta|Y) \propto p(Y|\theta)p(\theta) = Likelihood * prior$$

where θ is the parameter of interest.

The parameter of interest in this problem is $\vec{\mu}$. So we are interested to derive a conditional distribution (posterior) of $p(\vec{\mu}|Y)$. The likelihood and the prior we derived in previous step may be used to find the proportionality for the posterior;$posterior \propto$ (1) $*$ (2):

$$p(\vec{\mu}|Y) \propto \exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i-\mu_i)^2)\ .\ \exp(-\frac{1}{2\sigma^2}\sum_{i=1}^{n-1}(\mu_{i+1}-\mu_i)^2)$$

$$\propto \exp(-\frac{1}{2\sigma^2}[(\mu_1-Y_1)^2+\sum_{i=2}^{n}(\mu_i-Y_i)^2+(\mu_i-\mu_{i-1})^2)], \qquad (3)$$

(3) is the general form of proportionality for the posterior. To drive the conditional distribution for each μ, we first find $p(\mu_1|\mu_{-1,Y})$ considering that the proportionality principle allows us to remove the terms which do not involve any functional dependence of μ which means that we are allowed to exclude the terms which do not contain μ:

$$p(\mu_1|\mu_{-1},\vec{Y}) \propto \exp(-\frac{1}{2\sigma^2}[(\mu_1-Y_1)^2+(\mu_1-\mu_2)^2])$$

Using Hint B the right hand sid is proportional to:

$$\exp(-\frac{1}{\sigma^2}(\mu_1-\frac{Y_1+\mu_2}{2})^2)$$

Hence,

$$p(\mu_1|\mu_{-1},\vec{Y}) \propto \exp(-\frac{1}{\sigma^2}(\mu_1-\frac{Y_1+\mu_2}{2})^2) \ \sim\ N(\frac{Y_1+\mu_2}{2},\ \frac{\sigma^2}{2}), \qquad (4)$$

Next we derive the distribution $p(\mu_n|\vec{\mu}_{-n},\vec{Y})$:

Again according to proportionality principle and the hint B we have:

$$p(\mu_n|\vec{\mu}_{-n},\vec{Y}) \propto \exp(-\frac{1}{2\sigma^2}[(\mu_n-Y_n)^2+(\mu_n-\mu_{n-1})^2)]$$

$$p(\mu_n|\vec{\mu}_{-n},\vec{Y}) \propto \exp(-\frac{1}{\sigma^2}(\mu_n-\frac{Y_n+\mu_{n-1}}{2})^2) \ \sim\ N(\frac{Y_n+\mu_{n-1}}{2},\ \frac{\sigma^2}{2}), \qquad (5)$$

And finally we find the conditional distribution $p(\mu_i|\vec{\mu}_{-i},\vec{Y})$ for $i=2,...,n-1$.

For the i^{th} μ there are three terms in the posterior proportionality derived as (3) which contain μ_i as there are mu_{i-1} and μ_{i+1} terms befor and after that. Hence we have:

$$p(\mu_i|\vec{\mu}_{-i},\vec{Y}) \propto \exp(-\frac{1}{2\sigma^2}[(Y_i-\mu_i)^2+(\mu_i-\mu_{i-1})^2+(\mu_{i+1}-\mu_i)^2])$$

Using hint C we conclude that

$$p(\mu_i|\vec{\mu}_{-i},\vec{Y}) \propto \exp(-\frac{3}{2\sigma^2}(\mu_i - \frac{Y_i+\mu_{i-1}+\mu_{i+1}}{3})^2) \sim N(\frac{Y_i+\mu_{i-1}+\mu_{i+1}}{3}, \frac{\sigma^2}{3}), \quad (6)$$

4. Use the distributions derived in Step 3 to implement a Gibbs sampler that uses $\vec{\mu}^0 = (0, ..., 0)$ as a starting point. Run the Gibbs sampler to obtain 1000 values of $\vec{\mu}$ and then compute the expected value of $\vec{\mu}$ by using a Monte Carlo approach. Plot the expected value of $\vec{\mu}$ versus X and Y versus X in the same graph. Does it seem that you have managed to remove the noise? Does it seem that the expected value of $\vec{\mu}$ can catch the true underlying dependence between Y and X?

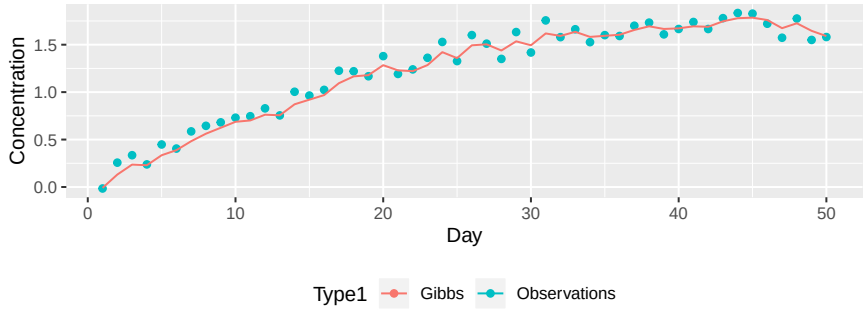
We are asked to obtain 1000 vectors, $\vec{\mu}$, each of which contains $(\mu_1, \mu_2, ..., \mu_n)$, then compute the expected value of each vector and plot the resulted mean vector as the expected values of concentration.

We construct a $1000*50$ zero matrix as the initial point. The we update each row implementing the equations $(4), (5), and (6)$ depending on the which column(which elemnt of the vectoe μ) we are updating:

```
y = chemic$Concentration
n= nrow(chemic)

M = 1000
Var = 0.2

mu = matrix(0, nrow = M, ncol = n)
set.seed(12345)
for (i in 1:M) {

  for (j in 1:n) {
    if(j ==1){
      mu[i,j] = rnorm(1, mean = (y[1]+mu[i,2])/2, sd = sqrt(Var/2))
      }else if(j == n){
        mu[i,j] = rnorm(1, mean = (y[n] + mu[i,n-1])/2, sd = sqrt(Var/2))
        }else{
          mu[i,j] = rnorm(1, mean = (y[j] + mu[i,j-1] + mu[i,j+1])/2,
                          sd = sqrt(Var/3))
        }
  }
}
```

Having computed all 1000 $\vec{\mu}$, we now compute the expected value of each vector and plot the resulted mean vector vs. day.

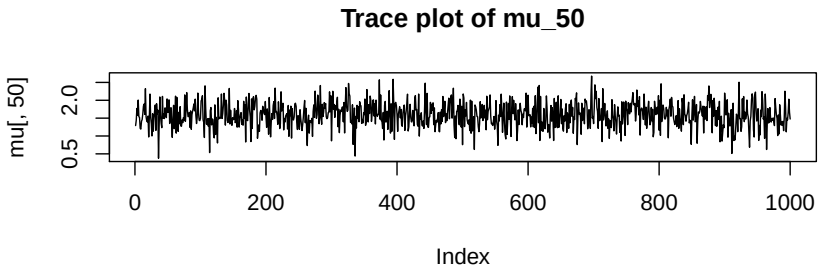
```
Mu.mean = apply(mu, 2, mean)
chemic$Mu = Mu.mean

chemic$Type1 = "Observations"
chemic$Type2 = "Gibbs"


ggplot(chemic)+ geom_point(mapping = aes(Day, Concentration, color=Type1))+
  geom_line(mapping = aes(Day, Mu, color = Type2)) + theme(legend.position = "bottom")
```



As the graph shows that the sampler seems to capture the relationship between Day and Concentration. Obviously it could remove the noise.

5. Make a trace plot for μ_n and comment on the burn-in period and convergence.

```
plot(mu[,50], type = 'l', main = "Trace plot of mu_50")
```



The trace plot of μ_{50} can be seen in the figure above. It seems that it fluctuates around a mean value about 1.5. Ignoring some high jumps within the data, we may say that it converged. The trace plot seems that converged as it started and no obvious burn in part may be distinguished precisely.